YARN Service Registry

# 一、简介

YARN Service Registry用于解决下面两个问题：客户端如何和YARN部署的服务和组件交互；简化Hadoop Core Service的注册和查找过程，从而减少配置的参数，使core services很容易的清除

分布式应用服务的注册和查询，从Xerox's Grapevine Service开始就是一个很难解决的问题。下面介绍YARN部署的分布式应用的注册及如何设置binding信息来和这些应用交互。

## 1.1 定义

Service：Hadoop YARN集群中部署的应用，例如Apache HBASE、Hcatlog、Storm。这些Service可以使长时间运行或者短时间运行

Service class: Sevices type的名字，在registry中用于path，和DNS的命名规则兼容，例如org-apache-hbase,org-apache-hcatlog

Component:服务的分布式组件，例如HBASE MASTER、REGIONS及REST servers。

Service instance: 一个应用的单个实例。例如，HBase集群demo1

Comonent Instance: 一个服务中单component的实例，例如，主机rack1-server6上运行的HBASE Master

Endpoint: 一个service instance及component instance的绑定。例如，HBASE zookeeper bindings，Region Server的JMX端口，HBASE MASTER的Web UI，HBASE REST的REST API。Endpoint，可能是内部使用，也可以使外部使用（Service instance的Clients）

Service Record: 注册中描述的一个服务实例或者组件实例的一个记录，包括其endpoints列表。

YARN RM、AM、Container、Application

## 1.2 binding Problem

Hadoop YARN允许应用运行在Hadoop Cluster中。其中一些是批处理作业或者查询，这部分可以通过YARN提供的API进行管理。但是，YARN部署Tomcat Web Service或者Apache HBase集群这些长时间执行的服务实例，YARN根据这些服务的components配置及服务器可用资源在集群中进行部署。这些服务实例需要被Clients端访问。一般它们的IP被注册到DNS或者配置文件中，但是在YARN中部署的实例，并不能提前知道他们的hostname及端口，因此Clients很难动态的访问这些应用。

YARN的基本注册功能允许App Master注册web URL及IPC address，但是这些不足够，因为这些应用的endpoint都不能注册，例如App Master执行所需要的REST URLS、Zookeeper路径及tasks的endpoint。更难的是，YARN app启动是唯一的instance ID在不断的变化，很难通过映射来完成binding info的解析，甚至可能会映射到不存在的服务实例。

# 二、Use Cases

## 2.1 Service Name Example

核心的Hadoop Services，不管是静态还是动态部署，通过有权限的用户将服务写入到/services路径下

/services/hdfs

/services/yarn

/services/oozie

YARN-deployed服务属于用户

/users/joe/org-apache-hbase/demo1

/users/joe/org-apache-hbase/demo1/components/regionserver1

## 2.2 Registration Use Cases

1. Hadoop core service（HDFS），不是运行在Yarn中，注册后，可以用于服务的查找。注册的方式是HDFS自己注册，或者通过管理工具
2. Long-lived 应用，registers self，注册信息提供给客户端。注册信息存在时间比App Master时间长，可能存在的时间为service instance的部署时间
3. Component instances注册自己，发布内部绑定信息，例如JMX PORTS
4. YARN-deployted应用，可以绑定独立的service实例（不管是静态的还是动态的）。例如，Tomcat Web pool 绑定到动态HBase实例(/users/joe/services/hbase/demo1)
5. Component实例使用registry，绑定到App Master的内部endpoint，（周期的传输心跳）

但是当前Yarn Registry不支持下面的用例：

Short-lived Yarn Application自动注册，包括所有的containers，大概任务结束后注销containers。Short-lived applications会给注册带来很大的额外开销。所有的应用都有注册选项，但是不会设置成自动，而且应用开发者需要建议不注册short-lived containers。

## 2.3 Lookup Use Cases

1. 客户端可以查询一个动态部署的服务实例，该服务的用户、service class及instance名称已知，例如/users/joe/services/hbase/demo1，检索信息需要连接到服务端
2. 客户端查询静态部署的Hadoop服务，例如/services/hdfs
3. App Master可以列举所有已注册的components实例，查找到他们的JMX port，初始化Web UI，及提供这些endpoints的链接
4. 用于可以访问 private HBase Service instance，/users/joe/service/hbase/demo1
5. 用户可以连接到集群的HBase Service，/services/hbase
6. 用户可以查询到远端/net/cluster4/services/hdfs下的binding信息。注册信息包括remote文件系统的webhdfs://URL
7. 用户列举HBase Service实例

ls /users/joe/services/hbase

1. 查找到集群中所有的HBase Services

find -endpointField.api=org.apache.hbase

1. 以后可能支持，通过DNS查找服务

这些注册建议会支持应用注册他们的service endpoints及客户端可以定位他们。

# 三、Service Registry的关键前提

## 3.1 允许Service instance的动态注册

1. YARN deployed service 实例必须能够注册他们的bindings，而且clients能够查询到
2. 核心的Hadoop Service实例能够注册Service endpoints
3. 如果service移动或者HA fails over,bings信息可以更新
4. service instances能够发布服务的不同endpoints，Web UI、RPC、REST、ZOOKEEPER等。而且必须能够注册证书及其他公共安全信息，这些信息作为binding的一部分

## 3.2 Registry Service Properties

1. registry高可用
2. 规模，在大集群中serivces及clients规模大，这限制了service可以发布的数据
3. 普遍性，不管是物理机、虚拟机及云中，Yarn集群都可以使用
4. 支持分层次的namespace及names，命名规则与DNS相对应，可以通过DNS协议访问
5. Registry API语言及协议
6. 跨语言，不依赖某种语言。Client Language与Service语言不同
7. 读取注册数据的REST API

## 3.3 访问控制

1. 可读
2. 限制写，防止非法操作

提供远程访问，支持通过Apache Knox或者云环境主机的访问

## 3.4 非必要需求

1. 注册并不是为了Liveness监测、leader-eletction及执行其他需要协商一致性的应用操作，而是为了在components 实例间分享binding信息
2. 注册并不是为了存储随机的应用状态，或者发布配置信息，而是发布service或者components提供的binding information的endpoints。这种应用可能会使registry（zookeeper）负载过重

# 四、Architecture

Apache Hadoop提供了基本的registry服务，注册信息为<string-names,records>，这些信息用于描述Service和Component instances。使用ZK作为基本的组件，选择部分ZK 命名空间作为service注册的根目录(默认是yarnRegistry)

这个基本实现中，YARN Service会用到Registry service API及Naming convention。注册信息，通过registry API而不是直接通过ZK来访问（ZK仅是一种实现）。

1、service的注册，绑定的path对应的值称为Service Record。路径是层次结构，/作为其根目录。

2、Service records作为持久的znodes注册到ZK中。这保证service中断后，record仍然存在（假设Client Code是弹性且易变的）

3、每个Service instance的service record罗列了service暴露出的多种协议的endpoints

4、每个endpoint协议，包括一下面部分

1）Protocol name包括：Web/REST/IPC及Zookeeper，类型为String

2）address，为了定位到endpoint需要specific details

3）address type，bind string的格式(URL,ZK Path，hostname:Port)。需要提前定义，例如protocal=REST，则binding type为URL,protocol=IPC则bidng type使用的address type为host/port

4）API，endpoint提供的API，Application specific，例如org.apache.hadoop.namenode,

org.apache.hadoop.webhdfs

5、Endpoints可能是对外的，更多是为编程而不是serivce 自己使用。internal，service内部交流。在Service Record中External及Internal的endpoints要区别对待

6、核心服务使用下面的定义规则：/service/{servicename}，例如/service/hdfs

7、Yarn Services应该按照下面的规则进行注册

/users/{username}/{serviceclass}/{instancename}

8、Components 实例，应该注册到下面的目录

/users/{username}/{serviceclass}/{instanceanme}/components/{componentname}

遵守这些规则的服务，必须有唯一的service class名字。每个component instance的名字必须唯一，可以使container ID。唯一的名字保证了service instance和components的路径唯一，而且可以根据特定的service class的所有实例列举service class路径下的所有children

# 五、Registry Model

Service entris必须持久化，Yarn和其他工具决定这些Service entry是否需要被删除。

## 5.1 Path Elements

在hostname path下的所有的path Elements必须符合RFC1123定义。常规expression：

([a-z0-9]|([a-z0-9][a-z0-9\-]\*[a-z0-9]))

这个规则保证了Registry heiarchy可以被DNS服务使用，而且所有的service class及names有效。

当用户名中出现空格/high unicode及其他字符时，会增加处理难度。这些paths需要转换成有效的DNS hostname entries

## 5.2 Service Record

Service Record有一些基本的信息，可能没有internal 及external endpoints，其包括下面的一些基本信息及两类endpoints（用户使用的external及app内部的列表）

|  |  |
| --- | --- |
| Name | Description |
| type: String | Always: "JSONServiceRecord" |
| description: String | Human-readable description. |
| external: List[Endpoint] | A list of service endpoints for external callers. |
| internal: List[Endpoint] | A list of service endpoints for internal use within the service instance. |

type必须是JSONServiceRecord。如果缺少这个字段，不能通过JSON parser来处理。

## 5.3 YARN Persistence Policies

当Application、attempt及container结束时，RM完成Service Record的清除工作。This allows service to register entries which have a lifespan bound to one of these aspects of YARN applications’ lifecycles.

这个特性仅当RM配置hadoop.registry.rm.enabled时才支持。当配置为true时，RM会根据下面的规则清理Service Record，disabled则不提供registry support。yarn:id及yarn:persistend定义了哪类record及child entris在components完成时可能被删掉。

yarn:id字段定义了匹配的app,attempt及container ID。yarn:persistence性质定义了record cleanup的trigger，implicityly定义yarn:id区域的类型。

使用yarn:的attributeds依赖实施该Policy的YARN layer。如果registry run standalone，则所有的记录默认持久化。

|  |  |  |
| --- | --- | --- |
| Name | Description | contents of `yarn:id` field |
| permanent | The record persists until removed manually. | (unused) |
| application | Remove when the YARN application defined in the id field terminates. | application ID |
| application-attempt | Remove when the current YARN application attempt finishes. | application attempt ID |
| container | Remove when the YARN container in the ID field finishes | container ID |

这个规则定义了当app，app attempt及container停止时，需要yarn:id字段匹配可以清理的app,attempt,container。如果id设置错了，那么就不会进行clean up操作，设置了不同的application及container，那么根据app的生命周期进行处理。

下面是Endpoint的定义：

|  |  |
| --- | --- |
| Name | Description |
| api: URI as String | API implemented at the end of the binding |
| protocol: String | Protocol. Examples: `http`, `https`, `hadoop-rpc`, `zookeeper`, `web`, `REST`, `SOAP`, ... |
| addressType: String | format of the binding |
| addresses: List[Map[String, String]] | a list of address maps |

所有的string fildls都有大小的限制，在text 描述中尽量不要用hiding complex json结构。

addressType:Address Type

定义了entries的格式，使用separare types后，一些工具例如Web viewer不需要识别protocol就可以处理binding strings。

|  |  |
| --- | --- |
| format | binding format |
| uri | uri:URI of endpoint |
| hostname | hostname: service host |
| inetaddress | hostname: service host, port: service port |
| path | path: generic unix filesystem path |
| zookeeper | hostname: service host, port: service port, path: ZK path |

在zookeeper binding中，每个entry都代表了quorum中的单个节点，hostname和port 定义了zk instance是的hostname及监听的端口。Path field列举了app使用的Zookeeper path。例如，HBASE可能会使用HBase Cluster的包含信息的znode。

在address列表中Path是必须要配置的元素，这保证了单个address包含足够的信息，可以连接到quorum及相关的Znode。可以定义新的Address type，非标准的type，使用x-作为前缀。

Filed api:Api Identifier

API包括URI，标识Endpoint的URI,必须唯一。API URI的定义有几下规则：

1. WSDL定义的服务，使用SOAP/WS-\*的URL
2. svn/git的URL管理了REST API定义的文档
3. classpath schema后面要跟随app package的路径
4. uuid，为产生的UUID

一般都使用标准的API URIS，有两种非标准的APIS，分布是http://，用户使用的Web site，classpath:javax.management.jmx，支持JMX管理协议（基于RMI）的endpoint。

## 5.4 Service Entries的例子

下面是YARN中部署的 tomcat应用，App创建及注册后，Registry信息如下所示：

/users

/devteam

/org-apache-tomcat

/test1

/components

/container-1408631738011-0001-01-000002

/container-1408631738011-0001-01-000001

/users/deveam/org-apache-tomcat/tomcat-test service record描述了总体的应用，loadbalancer的url导出后，如下所示：

{

"description" : "tomcat-based web application",

"external" : [ {

"api" : "http://internal.example.org/restapis/scheduler/20141026v1",

"addressType" : "uri",

"protocol" : "REST",

"addresses" : [

{ "uri" : "http://loadbalancer/" },

{ "uri" : "http://loadbalancer2/" }

]

} ],

"internal" : [ ]

}

Serive instance包括了两个component实例，每个用containerID转化后的DNS-compatible hostname来描述。每个Entries marked as ephemeral。如果entries在container中定义，当container被释放或者component失败，Entries会被自动消除。持久性的policy被设置为3。yarn:id域标志了container，完成后会触发entry的删除。

/users/devteam/org-apache-tomcat/test1/components/container-1408631738011-0001-01-000001

{

"yarn:id" : "container\_1408631738011\_0001\_01\_000001",

"yarn:persistence" : "container",

"description" : "",

"external" : [ {

"api" : "http://internal.example.org/restapis/scheduler/20141026v1",

"addressType" : "uri",

"protocol" : "REST",

"addresses" : [{ "uri" : "rack4server3:43572" } ]

} ],

"internal" : [ {

"api" : "classpath:javax.management.jmx",

"addressType" : "host/port",

"protocol" : "rmi",

"addresses" : [ {

"host" : "rack4server3",

"port" : "48551"

} ]

} ]

}

Component列举了他们的endpoint，REST API作为外部的endpoint，JMX address为Internal endpoint，如下所示：

/users/devteam/org-apache-tomcat/test1/components/container-1408631738011-0001-01-000002

{

"registrationTime" : 1408638082445,

"yarn:id" : "container\_1408631738011\_0001\_01\_000002",

"yarn:persistence" : "container",

"description" : null,

"external" : [ {

"api" : "http://internal.example.org/restapis/scheduler/20141026v1",

"addressType" : "uri",

"protocol" : "REST",

"addresses" : [ [ "http://rack1server28:35881" ] ]

} ],

"internal" : [ {

"api" : "classpath:javax.management.jmx",

"addressType" : "host/port",

"protocol" : "rmi",

"addresses" : [ {

"host" : "rack1server28",

"port" : "48551"

} ]

} ]

}

这个信息可能被load balancer用于列举componets，及创建component instances，接受传来的requests。同样，管理App可能列举所有的可用component实例及JMX port，通过这个端口获取监控信息。

# 六、Registry API

这里介绍Java app使用的Registry API，API是ZK操作的简化，基本操作包括build、paths、reading、writing，updating Entries及列举children。在Server中实现的REST API。只列举了部分异常。

所有的写操作必须假定可以与Registry service通信，与Zookeeper Client的看到的一致。Read-only获取的数据可能是过期的数据。所有的客户端都必须知道识别registry is a shared resource，在操作中会发生变化。

接口类为：RegistryOperations及RegistryPathStatus

# 七、安全性

Registry允许Service instance在有权限的情况下注册到路径下。

# 八、完整的Service Record例子

<http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/registry/yarn-registry.html>

上面的链接是完整的Service Record例子，发布一些endpoints，包括internal及external。

External:

IPC hostanme及port，用于client-am通信

AM web UI的URL

在app service Web UI下的一系列的REST URLS

Internal:

App部署时，AM为Containers提供的两个URLS(REST APIS)

Python agents运行在containers中，通过internal URLS与AM通信。Containe setup时record被resovled及缓存直到发生错误。